

НАСТРОЙКА ОБМЕНА ДАННЫМИ ПО ПРОТОКОЛУ НАRT НА КОНТРОЛЛЕРАХ СЕРИИ REGUL RX00

Руководство пользователя

DPA-302.3 Версия документа 1.7 Версия ПО 1.6.5.0 Июль 2021

Версия руководства пользователя	Описание изменения					
1.6	Добавлена история изменений руководства пользователя.					
	обавлены знаки с предупреждающей и поясняющей информацией.					
	<i>Раздел «Алгоритм работы HART-мастера»:</i> дополнено описание режима работы в зависимости от типа модуля.					
	Дополнительно по тексту внесены небольшие изменения с уточняющей информацией					
1.7	Добавлен новый раздел: «Поддержка устройств в Burst режиме».					
	Дополнительно по тексту внесены небольшие изменения с уточняющей информацией					

История изменений руководства пользователя

АННОТАЦИЯ

Настоящий документ содержит сведения о настройке передачи данных с применением протокола HART на промышленных логических контроллерах серии Regul RX00. Настройка осуществляется с помощью программного обеспечения Epsilon LD.

Данное руководство предназначено для эксплуатационного персонала и инженеровпроектировщиков АСУ ТП, которые должны:

- иметь, как минимум, среднее техническое образование;
- приступить к работе только после изучения данного руководства.

Обновление информации в Руководстве

Производитель ООО «Прософт-Системы» оставляет за собой право изменять информацию в настоящем Руководстве и обязуется публиковать более новые версии с внесенными изменениями. Обновленная версия Руководства доступна для скачивания на официальном сайте Производителя: https://www.prosoftsystems.ru/.

Для своевременного отслеживания выхода новой версии Руководства рекомендуется оформить подписку на обновление документа. Для этого необходимо на сайте Производителя: https://www.prosoftsystems.ru/ во вкладке «Документация» под иконками документов кликнуть на кнопку «Подписаться на обновления» и оставить свои контактные данные.

В руководстве присутствуют знаки с предупреждающей и поясняющей информацией. Каждый знак обозначает следующее:

ПРЕДУПРЕЖДАЮЩИЕ ЗНАКИ



ВНИМАНИЕ!

Здесь следует обратить внимание на способы и приемы, которые необходимо в точности выполнять во избежание ошибок при эксплуатации или настройке.

ИНФОРМАЦИОННЫЕ ЗНАКИ



ИНФОРМАЦИЯ

Здесь следует обратить внимание на важную информацию

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
Общие сведения	5
Перечень рекомендуемых документов	5
НАСТРОЙКА РАБОТЫ	6
Перечень действий при настройке HART	6
Алгоритм работы HART-мастера	7
ДОБАВЛЕНИЕ И НАСТРОЙКА УСТРОЙСТВ, РАБОТАЮЩИХ ПО	
ПРОТОКОЛУ HART	9
Добавление устройств и объектов в конфигурацию контроллера	9
Добавление HART-устройств к модулям аналогового ввода/вывода	10
Добавление HART-устройств к коммуникационным модулям	12
Добавление HART-устройств непосредственно к модулю центрального	
процессора	14
ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЬСКИХ КОМАНД, ПЕРЕДАВАЕМЫХ	ПО
HAK I	15
Добавление контейнеров HartDevice	15
Использование предустановленного набора команд	17
Дизайн собственных команд	18
Редактор hart команды	18
Библиотека PsIoDrvHartMaster	22
ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ И ПРИВЯЗКА ПЕРЕМЕННЫХ К	
КОМАНДАМ	24
Объявление переменных	24
Привязка переменных программы к командам	24
ОБРАБОТКА ВЫПОЛНЕНИЯ КОМАНДЫ	28
ПОДДЕРЖКА УСТРОЙСТВ В BURST РЕЖИМЕ	30
ФУНКЦИОНАЛЬНЫЙ БЛОК HARTUSERREQUEST И ДИНАМИЧЕС	КИ
ФОРМИРУЕМАЯ КОМАНДА	32
Общее описание	32
Объявление структуры данных команды непосредственно в коде	33
Использование команды из контейнера команд HART-устройства	35
Пример выполнения команды 34 – Write Primary Variable Damping Value	36

введение

Общие сведения

Программное обеспечение контроллера позволяет сконфигурировать его в качестве HART-мастера (по умолчанию – primary) и опрашивать Slave-устройства (интеллектуальные датчики), либо управлять ими (исполнительные устройства с поддержкой HART) по последовательной линии по протоколу HART 6.

Протокол HART основан на методе передачи данных с помощью частотной модуляции (Frequency Shift Keying, FSK), в соответствии с коммуникационным стандартом Bell 202. Цифровая информация передаётся частотами 1200 Гц (логическая 1) и 2200 Гц (логический 0), которые накладываются на аналоговый токовый сигнал. Скорость передачи данных для HART составляет 1,2 кбит/с.

Работа осуществляется в двух режимах:

- Одноточечный режим соединение «точка-точка», при этом протокол допускает параллельную работу двух мастеров (например, первичный-стационарный контроллер и вторичный- мобильный ручной коммуникатор);
- Многоточечный режим объединение в сеть нескольких ведомых устройств и двух мастеров. При этом по линии осуществляется только цифровая связь. Только одно из ведомых устройств на шине может работать в особом режиме ускоренной передачи (burst mode), в котором оно периодически отправляет в сеть пакет – ответ на заданную команду.

Начиная с версии СПО 1.6.5.0, реализован драйвер на уровне операционной системы (Hart Master OS), с поддержкой протокола HART 7 и технологии FDT/DTM. Подробное описание подключения к контроллеру по спецификации DTM приведено в документе «Настройка и работы REGUL GW DTM. Руководство пользователя».

Перечень рекомендуемых документов

Для получения информации по настройке других параметров контроллеров серии Regul RX00 в среде разработки Epsilon LD рекомендуется ознакомиться со следующими документами:

- Программное обеспечение Epsilon LD. Руководство пользователя;
- Regul R600. Системное руководство;
- Regul R500. Системное руководство;
- Regul R200. Системное руководство.

НАСТРОЙКА РАБОТЫ

Установите на компьютер программное обеспечение Epsilon LD. Описание процесса установки программы, а также инструкции по работе с программой приведены в документе «Программное обеспечение Epsilon LD. Руководство пользователя». Программа установки и документация доступны на сайте www.prosoftsystems.ru.

Запустите программу **Epsilon LD**. Откройте проект, в котором требуется настроить контроллер для обмена данными по протоколу HART. Если такого проекта нет, создайте его с помощью **Мастера конфигурации Regul** (описание приведено в разделе «Основные понятия среды разработки. Проект» документа «Программное обеспечение Epsilon LD. Руководство пользователя DPA-302»).

Перечень действий при настройке HART

Настройка НАRT начинается с добавления в проект НАRT-мастера (Hart Master), к которому, в свою очередь, должны быть добавлены конечные устройства Hart Outer Slave, непосредственно осуществляющие обмен данными по HART-протоколу. При использовании в проекте модулей ввода/вывода с поддержкой HART (см. соответствующее описание модуля в «Regul RX00. Системное руководство») устройство HART-мастер (Hart Master) добавляется непосредственно к модулю ввода/вывода (см. раздел «Добавление НАRT-устройств к модулю аналогового ввода/вывода») и ассоциируется с выбранным каналом этого модуля, а каждый экземпляр Hart Outer Slave на этом канале обозначает отдельное устройство (датчик) в сети.

В случаях, когда требуется обмен данными с HART-устройствами через внешний HART-модем, имеющий последовательный интерфейс, либо с устройствами, использующими только цифровую часть протокола HART по последовательному каналу, возможны следующие подключения HART-мастера:

- в проект добавляется коммуникационный модуль с нужным количеством последовательных портов (Extended Regul Serial Port), к каждому из которых может быть подключен HART-мастер (см. раздел «Добавление HART-устройств к коммуникационным модулям»);
- НАRТ-мастер добавляется непосредственно к последовательному порту (Regul Serial Port) модуля центрального процессора (см. раздел «Добавление HART-устройств непосредственно к модулю центрального процессора»).

Далее необходимо создать в проекте объекты типа **HartDevice**. Это контейнеры, содержащие HART-команды для slave-устройства. Следующий шаг – добавление команд в контейнер (см. раздел «Добавление пользовательских команд, передаваемых по HART»).

В МЭК-приложении следует объявить переменные соответствующего типа для каждой команды в контейнере HartDevice. Далее необходимо выполнить привязку контейнера с

командами к slave-устройству и привязку объявленной переменной к параметрам команды (см. раздел «Объявление переменных и привязка переменных к командам»).

Для выполнения HART-команды в МЭК-приложении организуется цикл обработки с анализом статуса команды. И, при переходе команды из фазы выполнения (InProcess) в состояние завершения команды (с ошибкой или без), производится обработка полученных данных. Значение статуса *Ok* устанавливается при успешном получении ответа от устройства и его обработке без ошибок (см. раздел «Обработка выполнения команды»).

Алгоритм работы HART-мастера

Одноточечный режим

Если на один вход/выход аналогового модуля с поддержкой HART подключено одно HART-устройство, то одновременно доступно получение как самого токового значения с датчика (его Primary Value, PV), так и обмен данными с устройством по HART-протоколу. При этом в устройстве, согласно спецификации HART (например, с помощью команды 6), рекомендуется задать адрес 0 и включить режим использования токового сигнала (заводские настройки по умолчанию).

При этом протокол допускает параллельную работу двух мастеров:

- первичное ведущее устройство(Primary);
- вторичное ведущее устройство (Secondary).

Многоточечный режим

Если на один вход/выход подключено несколько HART-устройств, то им назначаются уникальные адреса на шине и токовый сигнал переводится в минимально необходимое для функционирования устройства значение (4 мА). При этом работает только цифровая часть HART-протокола — обмен командами. Дополнительно, на уровне Hart_Outer_Slave осуществляется синхронизация доступа к одному каналу модуля — после захвата и выполнения одной попытки запроса (успешной или с ошибкой) объект Hart_Outer_Slave гарантированно освобождает канал на заданный период времени (100 мс).

ИНФОРМАЦИЯ

Определенные модули ввода/вывода контроллеров серии Regul RX00, оборудованные отдельным HART-модемом на каждый канал, позволяют подключать на выбранный канал до 10 HART-устройств.

В случае, если в модуле один HART-модем с помощью мультиплексирования обслуживает группу каналов, то, для оптимизации обмена данными, рекомендуется подключать на один канал модуля не более одного HART-устройства.

Согласно спецификации HART поддерживается арбитраж шины со вторым мастером и поддержка устройств, работающих в Burst режиме.

Режим работы каналов модуля

Если каждый канал модуля работает через собственный HART-модем, то выполнение обмена данными по протоколу HART по всем каналам происходит параллельно и независимо от соседнего канала. При этом обработка всех подключенных на один канал устройств с поддержкой HART производится последовательно, по одной команде, с синхронизацией доступа к HART-модему.

Если же каналы в модуле разделены на группы и каждая группа работает через один HART-модем, то производится двухуровневая синхронизация доступа. На первом уровне доступ к HART-модему последовательно пытаются получить экземпляры устройств **Hart Master**, добавленные в дерево устройств к соответствующим каналам модуля в рамках одной подгруппы каналов (например, каналы 1-8 и 9-16 в модуле R500 AI 16 081). Далее, на втором уровне, обработка всех подключенных на один канал устройств с поддержкой HART производится последовательно, по одной команде, также с синхронизацией доступа к HART-модему.



ИНФОРМАЦИЯ

Рекомендуется производить подключение конечных HART-устройств с равномерным распределением по группам каналов модуля и по каналам модуля в рамках одной подгруппы

ДОБАВЛЕНИЕ И НАСТРОЙКА УСТРОЙСТВ, РАБОТАЮЩИХ ПО ПРОТОКОЛУ HART

Добавление устройств и объектов в конфигурацию контроллера

Общий принцип добавления устройств/объектов в конфигурацию контроллера описан в разделе «Описание интерфейса. Добавление объектов» документа «Программное обеспечение Epsilon LD. Руководство пользователя». Далее добавляют соответствующие HART-устройства (Рисунок 1)



Рисунок 1 – НАКТ-устройства

 Image: Display the state of the state

Добавление HART-устройств к модулям аналогового ввода/вывода

Добавьте в проект модуль аналогового ввода/вывода с поддержкой HART. К модулю можно добавить один или несколько HART-мастеров (Hart Master), каждый из которых является контейнером для нескольких устройств Hart Outer Slave. (Рисунок 2) <u>Максимальное количество HART-мастеров</u> определяется количеством каналов в модуле.

🖹 \iint Regul_Bus (Regul Bus)	🗊 Добавить устройство	×
	Ш Добавить устройство Има: Hart_Master @ Добавить устройство Вставить устройство Подключить устройство Обновить устройство Строка для полнотекстового поиска Производитель: «All vendors> / Има Производитель Версия Описание: @ @ Hart @ @ Hart Master Prosoft-Systems"Ltd. 1.6.5.0 Устройство, которое работает как Hart настер на 16 канальных модулях. @ Hart Master O Prosoft-Systems"Ltd. 1.6.5.0 Устройство, которое работает как Hart настер на 16 канальных модулях.	
 ща дооваление ооъекта Добавить палку Добавить устройство Вставить устройство Откл. устройство Обновить устройство 	 <	ыть

Рисунок 2 – Добавление Hart Master

Далее к устройству **Hart Master** нужно подключить одно или несколько внешних slave-устройств (outer slaves), которые будут опрашиваться контроллером (*Regul* \rightarrow *Hart* \rightarrow *Hart Master* \rightarrow *Hart Outer Slave*) (Рисунок 3).

Regul_Bus (Regul Bus)	🇃 Добавить устройство			×
🗏 📕 0 : R500_ (R500 Крейт)				
ST_xx_x1x (ST xx x1x)	Имя: Hart_Outer_Slave			
PP_00_xxx (PP 00 xxx)	Лействие			
🗊 * : CU_00_xxx (CU 00 xxx)		× 0-	× 0.05	
🚔 🗊 AI_16_081 (AI 16 081)	• Добавить устроиство Вставит	ь устроиство О Подкл	почить устроиство О Обновить	устроиство
1: Hart_Master_Master}			c All completes	
2:Hart	Строка для полнотекстового поиска	Производитель:	<all veridors=""></all>	
💮 3 : Hart 🟥 Копировать	Имя	Производитель	Версия Описание:	
— 👔 4: Hart_I 💼 Вставить	E- 🖸 Regul			
5: Hart X Удалить	🖻 💮 Hart			
6 : Hart_	Hart Master			
т [] 7: Hart_	Hart Outer Slave	"Prosoft-Systems" Ltd.	1.6.4.30 Внешнее устройство	(датчик
— 🚹 8 : Hart_ 😭 Свойства	_			
9: Hart 1				
10 : Hart				
— 👔 11 : Hart 🗀 Добавить папку				
— 🔟 12 : Hart Добавить устройство	🔽 Группировать по категориям 🔲 От	тображать все версии (для	экспертов) 🗌 Показать устаре	вшие версии
— 🛐 13 : Hart Откл. устрейство				
— 🚹 14 : Hart Обновить устройство				
15: Hart	Имя: Hart Outer Slave Производитель: "Prosoft-Syste	ms" Ltd.	^	
16 : Hart	Группы: Hart Master			
Редактировать ооъект в	Версия: 1.6.4.30			
	Описание: Внешнее устройство	(датчик или испольнитель	ное	ے ا
	VCTDOЙCTBO) C DOB DEDWYOŇ Hart			
	Лобавить выбранное устройство ка	к последнего потомка		
	Hart_Master_5			
		со, пока окно открыто.)		
			Лобавить устройство	Законть
			Дооврить устроиство	Darbelle

Рисунок 3 – Добавление устройств Hart Outer Slave

Двойным щелчком по названию устройства **Hart Master** откройте вкладку параметров (Рисунок 4).

Hart_Master	×
Настройка Hart Master 🏾 罩	Hart Master IEC Objects
Настройки Hart Master	
Отладочный режим Вторичный мастер Номер канала в модуле	
<	>

Рисунок 4 – Параметры устройства Hart Master

Для настройки доступны следующие параметры:

- Отладочный режим установка флажка в этом поле включает режим добавления в журнал контроллера отладочных сообщений HART-мастера;
- Вторичный мастер установка флажка в этом поле задает тип мастера как Вторичное ведущее устройство (Secondary HART Master). По умолчанию устройство работает как Первичное ведущее устройство (Primary HART Master);
- **Номер канала в модуле** номер канала в модуле, к которому подключен и по которому будет опрашиваться устройство Hart Outer Slave.



ИНФОРМАЦИЯ

При подключении HART-мастера по последовательному порту не требуется выбор канала в модуле, поле не отображается в редакторе HART-мастера

Двойным щелчком по названию устройства **Hart Outer Slave** откройте вкладку параметров (Рисунок 5).

Hart_Outer_Slave	× •							
Настройка Hart OuterSlave	Hart команды 🛛 🗮 Regul Hart Out							
Общие параметры устро	Общие параметры устройства							
Отладочный режим								
Адрес слейва	0							
Таймаут слейва	1000							
Поведение в режиме СТОГ	Нет активности 🗸							
Длина преамбулы	8							
Количество попыток	4							
Burst режим								
<	>							

Рисунок 5 – Параметры устройства Hart Outer Slave

Во вкладке Hacтройка Hart Outer Slave доступны следующие параметры:

- Отладочный режим установка флажка в этом поле включает отладочный режим с трассировкой в журнал контроллера;
- Адрес слейва адрес конечного устройства (0-15). Используется в универсальной команде идентификации (Universal Command 0) при начальном опросе устройств на шине;
- Таймаут слейва максимальный таймаут ожидания ответа от HART-устройства, мс;
- **Поведение в режиме СТОП** указывает, что делать, если переключатель RUN/STOP модуля центрального процессора переведен в положение *STOP*. Возможные значения:
 - о Нет активности прекращение опроса,
 - о Нормальная работа продолжение работы в обычном режиме,
- Длина преамбулы посылка группы байт 0xFF перед каждой командой, необходимая для синхронизации приемников на шине. Для команды Universal Command 0 всегда используется 20 символов преамбулы;
- Количество попыток число попыток выполнения каждой команды;
- Вurst режим установка флажка в поле указывает на присутствие устройства на канале в Burst режиме. При попытке добавить еще одно устройство в Burst режиме, на вкладке Настройка Hart Outer Slave высветится знак ¹
 Наведя курсор на знак, появится предупреждающая информация: «Burst режим уже установлен для другого устройства».

Добавление HART-устройств к коммуникационным модулям

Добавьте в крейт коммуникационный модуль RX00 CP XX 011. К нему добавьте объект виртуального последовательного порта **Extended Regul Serial Port**. Предусмотрена возможность добавления нескольких портов расширения.

К каждому порту коммуникационного модуля следует добавить один **Hart Master**. Далее к устройству **Hart Master** нужно подключить одно или несколько внешних slave-устройств (outer slaves), которые будут опрашиваться контроллером (*Regul* \rightarrow *Hart* \rightarrow *Hart Master* \rightarrow *Hart Outer Slave*) (Рисунок 6). <u>Максимальное количество устройств</u> – 10.



Рисунок 6 – Добавление порта Extended Regul Serial Port и HART-устройств к модулю RX00 CP XX 011

В окне дерева устройств двойным щелчком по названию порта откройте главную вкладку параметров порта. Перейдите на внутреннюю вкладку Настройки последовательного порта (Рисунок 7). Номер порта выставляется автоматически (инкрементируется при добавлении нового устройства Extended Regul Serial Port).

♦ Hart_project.project - Epsilon LD	- 🗆 X
Файл Правка Вид Проект Компиляция Онлайн 管 🚅 🔚 🎒 🗠 🗠 👗 🗈 💼 🔁 🗙 🏘 🕼	Отладка Инструменты Окно Справка 📾 🛅 - 🔓 🕮 😋 🧐 🕨 🔳 🔏 🗊 - 🚍 🔄
Устройства 🗢 🔫 🗙	Extended_Regul_Serial_Port 🗙 👻
B Hart_project	Настройки последовательного порта Regul Serial Port
 	Номер порта: Порт 1 Порт 2 Порт 3 Порт 4
= 1: Extended Regul Serial Port (E	Скорость: 1200 ~
🗟 - 🔟 Hart_Master_1 (Hart Master)	Кол-во бит данных: 8 🗸
Hart_Outer_Slave_1 (Har	Четность: проверка на нечет 🗸
	Кол-во стоп-бит: 1 ~
Последняя компиляция: 📀 0 🕐 0 Предком	ил.: 🗸 Текущий пользователь: (никто)

Рисунок 7 – Настройки последовательного порта

Для HART установите следующие параметры:

- Скорость 1200;
- Кол-во бит данных 8;
- Четность проверка на нечетность;
- Кол-во стоп-бит по умолчанию 1.

Настройки устройств Hart Master и Hart Outer Slave аналогичны настройкам, описанным для модулей аналогового ввода/вывода.

Добавление HART-устройств непосредственно к модулю центрального процессора

Добавьте к головному устройству последовательный порт Regul Serial Port (Рисунок 8).

1	🗊 Добавить устр	ойство		×
	Имя: Regul_Ser Действие:	ial_Port	стройство <u>П</u> одключи	ить устройство О
	Устройство:			
Устройства х	Производитель:	<all vendors=""></all>		~
Hart_project REGUL_R500_71 (REGUL R500-71) REGUL_R500_71 (REGUL R500-71) Regul_Blas (Regul Bus)	Имя — П Regul — Моd — Моd — Мос — П Пос — П Пос — П Пос — Г Промыш — Отображать в — Группировать — Показать уста	bus 60870 iedoвательный порт Regul Serial Port iленные сети (fieldbus) все версии (для эксперто ь по категориям аревшие версии	Производитель "Prosoft-Systems"Ltd. ю)	Версия
	Информация: Мия: Regu Произво, Группы: I Версия: 1 Добавить выбра (Можно вы	ul Serial Port дитель: "Prosoft-System: Последовательный порт I.5.0.5 анное устройство как брать другой таргет-узе	s" Ltd. последнего потомка I л, пока окно открыто.) Добавить устройст	REGUL_R500_71 Во Закрыть

Рисунок 8 – Добавление последовательного порта

Предусмотрено добавление не более двух последовательных портов.

К последовательному порту следует добавить один **Hart Master**. Далее к устройству **Hart Master** нужно подключить одно или несколько внешних slave-устройств (outer slaves), которые будут опрашиваться контроллером (*Regul* \rightarrow *Hart* \rightarrow *Hart Master* \rightarrow *Hart Outer Slave*). <u>Максимальное количество устройств</u> – 10.

Настройки устройств Hart Master и Hart Outer Slave описаны в предыдущих разделах.

ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЬСКИХ КОМАНД, ПЕРЕДАВАЕМЫХ ПО НА<mark>R</mark>Т

Добавление контейнеров HartDevice

Все пользовательские команды, используемые при обмене данными по HART-протоколу, хранятся в специальных объектах **HartDevice**, являющихся контейнерами для команд. Это позволяет группировать команды для конкретных датчиков и других опрашиваемых устройств. Каждый контейнер может содержать множество различных команд. Но к каждому slave-устройству может быть привязан только один контейнер **HartDevice**.

Для добавления контейнера необходимо выполнить следующие действия:

в окне дерева устройств поставьте курсор на объект Application, правой кнопкой мыши вызовите контекстное меню, выберите Добавить объект... → HartDevice... (Рисунок 9);



Рисунок 9 – Добавление объекта в конфигурацию контроллера

 откроется окно Добавить HartDevice. В поле Имя укажите имя контейнера. Например, это может быть название датчика (опрашиваемого устройства) (Рисунок 10). Нажмите кнопку Добавить.

Добавит	ь HartDevice	X
	HartDevice	
Имя:	HartDevice_1	
		Добавить Отмена

Рисунок 10 – Добавление контейнера



ИНФОРМАЦИЯ

В дальнейшем при необходимости контейнер можно переименовать (в дереве устройств)

Объект типа HartDevice добавляется в дерево устройств. Автоматически открывается **Редактор hart-устройства** (Рисунок 11).

♦ Hart_project.project* - Epsilon LD		– 🗆 ×
Файл Правка Вид Проект Компиляция Онлай	н Отладка Инструменты Окно Справка	
🎦 🛩 🔚 🚳 🗠 🗠 👗 🖻 🛍 🗙 🛤 🕼	🛍 🛅 • 🖆 🛗 💜 💖 - 🕞 火 🗊 🖅 🛬 🕫	\$ ⇔ ≓'
Устройства 🗢 🕂 🗙	Редактор hart устройства 🗙	•
□ 🗿 Hart_project 🔹 💌	Импорт команд из другого hart устройства	
🖹 💮 🔟 REGUL_R500_71 (REGUL R500-71)		
Plc Logic	Найти все hart устройства	
Application	Выберите hart устройство: Выберите команди	ы: Выбрать все
Sensor_1_Hart		
Sensor_2_Hart	Выбрать Имя	Код
📶 Менеджер библиотек		
PLC_PRG (PRG)		
🗄 🌃 Конфигурация задач		
🗄 📺 Regul_Bus (Regul Bus)		
	Импортировать выбранные команды	
💥 Устройства 🔲 POU	<	>
	Последняя компиляция: 😳 0 😗 0 Предкомпил.: 🗸	Текущий пользовате

Рисунок 11 – Редактор hart устройства

Нажмите кнопку *Найти все hart устройства*. В поле **Выберите hart устройство**: отобразится список всех контейнеров типа HartDevice, имеющихся в текущем проекте и во всех подключенных библиотеках (Рисунок 12).

Редактор hart устройства × Імпорт команд из другого hart устройс	тва			
Найти все hart устройства				
Выберите hart устройство:	Выберите ко	манды:	Выбрать в	ce
Sensor 2 Hart	Выбрать	Имя	Код	^
DefaultHartDeviceRev7		Cmn_114	114	
		Cmn_113	113	
	·····	Cmn_109	109	
		Cmn_108	108	
Импортировать выбранные команд	ы	Cmn_107	107	
		Cmn_106	106	
		Cmn_105	105	
		Cmn_83	83	
		Cmn_82	82	
		Cmn_81	81	
	· 🗋	Cmn_80	80	
	···· 🗋	Cmn_79	79	
	···· 🗋	Cmn_76	76	
		Cmn_75	75	
		Cmn_74	74	
		Cmn_73	73	
	ļ	Cmn 72	72	\checkmark

Рисунок 12 – Редактор hart устройства

Объект с именем **DefaultHartDeviceRev6/7** – это контейнер с набором универсальных и общих команд, применяемых в спецификации HART Revision 6.0 и HART Revision 7.0 соответственно (для драйвера, реализованного на уровне операционной системы). Префикс **uni** означает, что это команда из спецификации универсальных команд HART (Universal Command Specification), **Cmn** – команда из спецификации общепринятых команд (Common Practice Command Specification). Контейнер является частью библиотеки PsIoDrvHartMaster, которая, в свою очередь, автоматически устанавливается вместе с пакетом, содержащим настройки для обмена по HART-протоколу. Таким образом, пользователю по умолчанию доступен контейнер, из которого можно импортировать нужные команды, и не создавать их вручную.

Использование предустановленного набора команд

Для использования предустановленного набора команд в **Редакторе hart устройства** в поле **Выберите hart устройство:** выберите, например, **DefaultHartDeviceRev6**. В правой части окна в блоке **Выберите команды:** отобразится список всех команд, имеющихся в данном контейнере. Установите флажки для нужных команд и нажмите кнопку *Импортировать* выбранные команды. Команды будут скопированы в пользовательский контейнер (Рисунок 13).



Рисунок 13 – Импорт команд

Для просмотра параметров команды щелкните дважды левой кнопкой мыши по названию команды. Откроется **Редактор hart команды** (Рисунок 14).

Редактор hart команды 🗙										
Параметры команды — Проверка формата структур										
Имя:	Cmn 33				Результат:	5 C (P) () (P	Ok			
HAPT Bencus:	<u></u>				- Требуемый форм	ат структуры перелачи:	[1 1 1 1 0]			
пакт верски	р ~				i pooyoninin + opin	ат отруктуры порода на	[1,1,1,1,0]			
Код:	33 📫				Требуемый форм	ат структуры приёма:	[1,1,4,1,1,4,1,1	,4,1,1,4,0]		
Описание:	Read Device Variable									
		-								
Формат стру	/ктуры передачи д	анных				Формат структ	/ры приёма да	нных		
Deferrer un	000 0000					Deferrer upper	0000			
дооавить но	Boe none					дооавить новое	поле			
Имя		Тип	Размер	Описание		Имя		Тип	Размер	Описание
Slot_0_De	evice_Variable_Code	BYTE	1	Slot 0 Devic	e Variable Code	Slot_0_Device	_Variable_Code	BYTE	1	Slot 0: Device Variable Code
Slot_1_De	evice_Variable_Code	BYTE	1	Slot 1 Devic	e Variable Code	Slot_0_Units_	Code	BYTE	1	Slot 0: Units Code
Slot_2_De	evice_Variable_Code	BYTE	1	Slot 2 Devic	e Variable Code	Slot_0_Device	Variable_Value	REAL	4	Slot 0 Device Variable Value
Slot_3_De	evice_Variable_Code	BYTE	1	Slot 3 Devic	e Variable Code	Slot_1_Device	_Variable_Code	BYTE	1	Slot_1_Device_Variable_Code
						Slot_1_Units_	Code	BYTE	1	Slot_1_Units_Code
						Slot_1_Device	_Variable_Value	REAL	4	Slot_1_Device_Variable_Value
						Slot_2_Device	_Variable_Code	BYTE	1	Slot_2_Device_Variable_Code
						Slot_2_Units_	Code	BYTE	1	Slot_2_Units_Code
					Slot_2_Device	_Variable_Value	REAL	4	Slot_2_Device_Variable_Value	
						Slot_3_Device	_Variable_Code	BYTE	1	Slot_3_Device_Variable_Code
						Slot_3_Units_	Code	BYTE	1	Slot_3_Units_Code
						Slot_3_Device	_Variable_Value	REAL	4	Slot_3_Device_Variable_Value

Рисунок 14 – Редактор hart команды

Дизайн собственных команд

Редактор hart команды

У пользователя может возникнуть необходимость создания контейнера с новыми командами, отсутствующими в подключенных библиотеках, например, при работе с версиями HART, отличными от ревизии 6.0/7.0. В программе предусмотрена возможность редактирования

существующих команд и создания собственных новых команд. Также можно разработать и предоставить новую библиотеку с контейнером, содержащим требуемые команды.

Для внесения изменений в существующую команду (импортированную или созданную вручную) щелкните дважды левой кнопкой мыши по названию команды. Откроется окно **Редактор hart команды**, где можно изменить код и описание команды, а также добавить/изменить поля в форматах структур приема/передачи данных (Рисунок 15).

При изменении формата структур для определенного диапазона кодов команд (это команды интегрированного в библиотеку HART-устройства DefaultHartDeviceRev6/7) предусмотрена автоматическая проверка формата структур команды по коду. Если формат структур не соответствует спецификации HART для данного кода команды, то в поле **Результат:** появляется сообщение об ошибке (Рисунок 15). При запуске проекта такая команда не будет выполняться.

📔 Редакто	ор hart команды 🗙											•
Параметры ком	анды				Проверка формат	формата структур						^
Имя: С	Cmn_34				Результат:		Ошибка в формате структуры передачи					
HART версия: е	НАRТ версия: 6 V			Требуемый формат структуры передачи: [4,0]								
Код: 3	34 🔹				Требуемый форма	ат структуры приёма:	[4,0]					
Описание:	Write Primary Variable Da	imping Valu	e									
Формат струк	туры передачи дан	ных				Формат структур	ы приёма данны	x				
Добавить ново	ое поле					Добавить новое п	оле					
Имя		Тип	Размер	Описани	e	Имя		Тип	Размер	Описание		
Primary_Var	riable_Damping_Value	REAL	4	Primary Va	riable Damping Value Primary_Variable_[_Damping_Value1	REAL	4	Primary Varia	ole Damping Valu	
var_name_2	2	LREAL	8									
												~

Рисунок 15 – Редактор hart команды

Для создания новой команды поместите курсор на название контейнера. Правой кнопкой мыши вызовите контекстное меню и выберите *Добавить объект* ⇒ *HartCommand*... Откроется окно **Добавить HartCommand**, где в поле **Имя:** задайте имя команды (Рисунок 16).

Добавит 📄	ъ HartCommand HartCommand		×
Имя:	HartCommand		
		Добавить	Отмена

Рисунок 16 – Добавление НАRТ-команды



ИНФОРМАЦИЯ

В дальнейшем при необходимости команду можно переименовать (в дереве устройств)

Нажмите кнопку *Добавить*. Откроется окно **Редактор hart команды** с пустыми полями (Рисунок 17).

🖉 Редактор hart команды 🗙		-
Параметры команды	Проверка формата структур	^
Имя: HartCommand	Результат: Ошибка в формате структуры приема	
НАRТ версия: 6 🗸	Требуемый формат структуры передачи: [0]	
Код: 0 🚖	Требуемый формат структуры приёма: [1,1,1,1,1,1,1,1,1,1,1,1,2,1,0]	
Описание:		
Формат структуры передачи данных	Формат структуры приёма данных	
Добавить новое поле	Добавить новое поле	
Имя Тип Размер Описание	Имя Тип Размер Описание	•

Рисунок 17 – Редактор hart команды при создании новой команды

В поле **HART версия:** выберите значение 6 или 7 из раскрывающего списка, согласно версии файлов описания и библиотек применяемой спецификации - HART Revision 6.0 или HART Revision 7.0 соответственно.

В поле **Код:** с помощью стрелок или вручную введите код команды – номер команды согласно спецификации HART, соответствующий ее функционалу. Произойдет автоматическая проверка команды по коду. Если команда с таким кодом есть в библиотеке, то в блоке **Проверка формата структур** появляется информация о требуемых форматах структур приема/передачи. Сообщение об ошибке присутствует на этом этапе, т.к. форматы пока не определены.

Если команда не описана в подключенных библиотеках, то в поле **Результат:** появится сообщение: «*Проверка структуры отключена*» (Рисунок 18). Для такой команды нет рекомендаций по форматам структур приема/передачи данных. Такая команда будет добавлена на исполнение.

Редактор hart	команд	ы										x
Параметры ко	манды					Про	в	ерка форма	та стр	уктур		
Имя:	HartCom	nmand2				Результат:				Проверка структуры отключена		
HART версия:	рсия: 6 ~					Требуемый формат структуры передачи: [1,1,0]						
Код:	од: 168					Tpe	бу	/емый форм	ат стр	уктуры приёма:	[1,1,0]	
Описание:	Изменен	ие адреса	устройства									
Формат стру	уктуры г	тередачи	данных			Формат структуры приёма данных						
Добавить но	вое поле	e						Добавить	ново	е поле		
Имя		Тип	Размер	Описание				Имя	Тип	Размер О	писание	
Polling_A	ddress	USINT	1	Polling Address of	fDevic	e						
Loop		BYTE	1	Loop Current Mr	4							
					Ň	опиро	DB	ать				
					В	ставит	ть	•				
					P	Редактировать						
					У	далит	ъ					

Рисунок 18 – Формирование структур приема/передачи данных

Нажмите кнопку *Добавить новое поле*. Откроется редактор декларации структуры (Рисунок 19).

Редактор декларации структуры									
Имя:	var_name_1								
Тип:	LREAL	∨ □ массив?	1						
Описание:									
Ok	Отмена								

Рисунок 19 – Редактор декларации структуры

Введите имя команды. Выберите тип поля в соответствии со спецификацией HART или вашими требованиями.

Необязательное поле Описание заполняется по желанию и несет описательную функцию.

Нажмите кнопку *OK*. В Редакторе hart команды появится созданное поле. В дальнейшем эти поля можно редактировать, удалять, копировать с помощью контекстного меню, вызываемого правой кнопкой мыши.

После того, как все поля добавлены/отредактированы, закройте **Редактор hart команды**. Новая команда с заданными параметрами будет добавлена в контейнер и будет отображаться в дереве устройств.

Библиотека PsIoDrvHartMaster

В библиотеке **PsIoDrvHartMaster** для каждой команды, указанной в списке пользовательских команд, определены структуры, где поля соответствуют блоку данных как в запросе, так и в ответе. Дополнительно каждая команда содержит четыре поля (Рисунок 20):

- CmdRespCode;
- CmdDevStatus;
- CmdStatus;
- CmdTrigger.



Рисунок 20 – Дополнительные свойства команды

CmdRespCode и **CmdDevStatus** – после получения ответа от HART-устройства в эти поля копируются два байта Response Code и Field Device Status, которые, согласно спецификации, содержатся в ответе на любую HART-команду; пользователь может самостоятельно проанализировать их содержимое с точки зрения получения расширенной информации о состоянии опрашиваемого устройства.

CmdStatus представляет собой перечисление текущего состояния команды, которое используется в МЭК-приложении для отслеживания цикла обработки команды:

- после добавления команды в очередь она получает статус *Idle* (находится в ожидании начала выполнения по таймеру или по триггеру);
- при переходе в фазу выполнения статус меняется на InProcess;
- при успешном получении ответа от устройства и его обработке без ошибок статус устанавливается в значение Ok; при этом в поле данных команды rdata содержится блок данных ответа от устройства и возможен доступ к его отдельным полям согласно структуре конкретной команды;
- в случае ошибки выполнения команды статус может принимать следующие значения:
 - *Timeout* от внешнего HART-устройства не был получен полный ответ на команду в течение периода Slave Timeout, заданного для текущего Hart_Outer_Slave;

- *ErrorInResponse* ошибка обработки полученного ответа (неверная длина или неверная структура ответа, неверный адрес, команда не реализована в устройстве, устройство занято, наличие ошибки в байте Response Code);
- UserReqArg ошибка параметров отдельной динамически формируемой команды, реализованной в функциональном блоке HartUserRequest библиотеки;
- о *InternalError* ошибка, связанная с обработкой команды внутри библиотеки;
- после выполнения команды и получения ответа ее статус не сбрасывается в *Idle*, а сохраняет свое значение до следующего начала цикла обработки и перехода в состояние *InProcess*;

Полное описание кодов перечисления доступно в МЭК-библиотеке *PsIoDrvHartMaster* (Таблица 1).

Код	Описание
Idle = 1	В ожидании
InProcess = 2	Идет выполнение цикла "запрос-ответ"
Ok = 3	Данные достоверны, команда успешно выполнена
Timeout = 4	Ошибка - нет ответа
ErrorInResponse = 5	Ошибка - ошибка в ответе
UserReqArg = 6	Ошибка - некорректный аргумент в пользовательском запросе
InternalError = 7	Ошибка - внутренняя ошибка обработки команды

Таблица 1 – Коды перечисления команды CmdStatus

CmdTrigger – управляющий флаг для команд по требованию (Trigger); в библиотеке фиксируются все переходы триггера из состояния *False* (0) в состояние *True* (1) и команда будет добавлена в очередь выполнения заданное число раз.

ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ И ПРИВЯЗКА ПЕРЕМЕННЫХ К КОМАНДАМ

Объявление переменных

Для того, чтобы HART-команды выполнялись, требуется добавить в программном коде переменные, представляющие из себя структуры определенных типов, описанных в библиотеке PsIoDrvHartMaster (PsIoDrvHartMaster_OS), и соответствующие добавленным в пункте выше командам для устройства Hart Outer Slave. Для всех добавленных HART-команд автоматически генерируются новые типы структур с именами, состоящими последовательно из имени контейнера (HartDevice), затем символа подчеркивания и имени переменной (HartCommand). Например, для контейнера с именем Metran_P_Sensor и командой Read_Primary_Variable будет сгенерирован тип структуры с именем Metran_P_Sensor_Read_Primary_Variable.

Для создания переменных откройте редактор ПЛК-программы (МЭК-приложение). Например, в редакторе ST для программы MAIN создание переменной выглядит следующим образом: *cmd: название контейнера_название команды*. (Рисунок 21).



Рисунок 21 – Пример объявления переменной

Привязка переменных программы к командам

Первым шагом следует привязать контейнер к slave-устройству, далее привязать переменные программы к командам. К одному slave-устройству допустимо привязать только один контейнер с командами.

Двойным щелчком по названию устройства **Hart Outer Slave** откройте редактор устройства. Перейдите на вкладку **Hart Команды** (Рисунок 22).

Hart_Outer_Slave	2 X							•
Настройка Hart OuterSlave	Hart команды	🗮 Regul Hart C	uter Slave IEC Objects	Состояние	🕕 Информация			
– Hart устройство								^
Привязанное устройств Выберите новое hart ус	о:							
		~	G Đ					
Команды:								
Выбрать Имя К	Код команды	Тип вызова	Время цикла (мс)	Переменн	ая для привязки	Описание	Burst режим	
				1				~

Рисунок 22 – Вкладка «Hart Команды»

Нажмите кнопку поиска устройства . В поле Выберите новое hart устройство: найдите в раскрывающемся списке название контейнера, выберите его. Нажмите кнопку . В поле Привязанное устройство: появится название контейнера, а в поле Команды: – список команд, которые содержит этот контейнер (Рисунок 23).

/ j Hart_O	Outer_Slav	e X						•
Настройка На	art OuterSlave	e Hartкоманды	Regul Hart Oute	er Slave IEC Objects Co	остояние 🍈 Информация			
-Hart устрой	йство							
Привязанн	ое устройст	BO: Sensor 1 Hart						
Rufapurra	 waaaa hart y							
высерите	новое пагт у	строиство:						
Sensor_1_	Hart		~	3 2				
K								
команды:				1				
Выбрать	Имя	Код команды	Тип вызова	Время цикла (мс)	Переменная для привязки	Описание	Burst режим	^
	Uni_0	0	Trigger	-	PLC PRG.dev1 cmd00	Read Unique Identifier		
···· 🗸	Uni_1	1	Trigger	-	PLC_PRG.dev1_cmd01	Read Primary Variable	\checkmark	
	Uni_2	2	Trigger	-	PLC_PRG.dev1_cmd02	Read Loop Current And Percent Of Range	\checkmark	
		2	Trigger	-	PLC_PRG.dev1_cmd03	Read Dynamic Variables And Loop Current		
	Uni_3	3	mgger			recar of frame variables what coop carrent		
	Uni_3 Uni_6	6	Trigger	-	PLC_PRG.dev1_cmd06	Write Polling Address		
	Uni_3 Uni_6 Uni_7	6 7	Trigger Trigger	-	PLC_PRG.dev1_cmd06 PLC_PRG.dev1_cmd07	Write Polling Address Read Loop Configuration		

Рисунок 23 – Контейнер привязан к slave-устройству

В строке нужной команды установите флажок в поле **Выбрать** и станут доступны для редактирования параметры команды.

В поле Тип вызова команды выбирается, как команда будет добавляться в очередь выполнения, при этом доступны следующие значения для настройки:

- о *заданный период (Timer)* период выполнения команды, мс. Для редактирования будет доступно поле **Время цикла**,
- по требованию (Trigger) для добавления команды в очередь выполнения. Во время выполнения МЭК-приложения каждый раз, когда значение этого поля для переменной, соответствующей добавленной команде, переходит из False в True команда добавляется в очередь,

В поле **Переменная** для привязки можно указать необходимую переменную. Имя переменной можно ввести вручную или через кнопку ..., открывающую окно **Ассистент** ввода. Раскрывая иерархический список, найдите нужную переменную (Рисунок 24).

	¬			
Переменные наблюдения	🔺 Имя	Тип	Адрес	Источник
	□ [] REGUL_R500_71			
	🖹 🔘 Application	Приложение		
	🖹 📄 PLC_PRG	PROGRAM		
	😟 🖗 cmd	Sensor_1_Hart_Cmn_34		
	🗄 🦓 IoConfig Globals	VAR GLOBAL		
		VAR_OLODAL		
✓ <u>С</u> труктурированный вид		MAL OLODAL		
Структурированный вид Показывать документацию окументация:	Вставка с	аргументами	Вставка	а с префиксом
Структурированный вид Показывать документацию окументация: cmd: Sensor_1_Hart_Cmn_1	Вставка с 34;	аргументами	Вставка	а с префиксом
	Вставка с 34;	аргументами	Вставка	а с префиксом
☐ Структурированный вид ☐ Показывать документацию окументация: cmd: Sensor_1_Hart_Cmn_; (VAR)	Вставка с 34;	аргументами	Вставка	а с префиксом

Рисунок 24 – Ассистент ввода

Выберите переменную для привязки. Окно **Ассистент ввода** закроется, а в строке команды в поле **Переменная** для привязки появится название переменной (Рисунок 25). Для сохранения поместите курсор в любое место окна.

art_Outer_	Slave							
астройка Н	lart OuterSlave	lart команды	Regul Hart Outer	Slave Конфигура	ация Состояние 🧃) Информ	ация	
Hart устро	йство							
Тривязанн	юе устройство:	Sensor 1 Hart						
Выберите	новое hart vcтро	йство:						
blocpine	noboe nare yerpo	incruo.			1			
Concer 1	Linet							
Sensor_1	_Hart		~	G Đ				
Sensor_1_ оманды:	_Hart		~	9 Đ				
Sensor_1 _. оманды: Выбрать	_Hart Имя	Код команды	Тип вызова	Время цикла	Переменная для п	тривязки	Описание	Burst режим
Sensor_1 оманды: Выбрать	_Hart Имя HartCommand	Код команды 4	✓ Тип вызова	Время цикла	Переменная для п	привязки	Описание	Burst режим
Sensor_1 оманды: Выбрать	_Hart Имя HartCommand Cmn_33	Код команды 4 33	✓ Тип вызова	Время цикла	Переменная для п	привязки	Описание Read Device Variables	Burst режим
Sensor_1 оманды: Выбрать	Hart Имя HartCommand Cmn_33 Cmn_34	Код команды 4 33 34	Тип вызова Timer	Время цикла	Переменная для п PLC_PRG.cmd	привязки	Описание Read Device Variables Write Primary Variable Damping Value	Burst режим
Sensor_1 оманды: Выбрать	Hart Имя HartCommand Cmn_33 Cmn_34 Cmn_35	Код команды 4 33 34 35	Тип вызова Timer	Время цикла 0	Переменная для п PLC_PRG.cmd	привязки	Описание Read Device Variables Write Primary Variable Damping Value Write Primary Variable Range Values	Burst режим

Рисунок 25 – Переменная для привязки

В поле **Burst режим** пользователь может установить флажок, указывающий, что данная команда будет использоваться для обработки Burst-пакетов, полученных мастером в сети от соответствующих устройств (см. раздел «Поддержка устройств в Burst режиме»).

К каждой команде можно привязать только переменную соответствующего типа. В случае несовпадения типов появится сообщение об ошибке (Рисунок 26).

♦ Hart_project.project* - Epsilon LD							- 🗆 ×	(
Файл Правка Вид Проект Компиляция Он	лайн Отлад	ка Инстру	менты Окно	Справка				
🎦 🚅 🖬 🕘 🗠 🗠 🐰 🖻 🛍 🗙 🕅	🕼 a·b 🛱	} ₩ - [ງ 🛯 🏙 🛛 😋 👒		(≣ f≣ d≊ +≣ \$	⇔ ☴'		
Устройства 🗸 🕂 🗙	Hart	Outer_Slav	ve 🗙					•
Hart_project	Настройка Н	lart OuterSlav	ve Hart команды	Regul Hart C	Outer Slave Конфигурация	Состояние	🚯 Информация	
🖻 🎲 🗊 REGUL_R500_71 (REGUL R500-71)	Hart umme?							
		ICTBO			1			
Application	Привязанно	ре устройств	HartDevice					
hart_data	Выберите н	ювое hart yo	стройство:					
= HartDevice					G Đ			
Cmn_33								
<u>i</u> Cmn_34	Команды:							
Hartuserkequest	Выбрать	Имя	Код команды	Тип В	Переменная для при	вязки Описа	ание	_
		Cmn 33	33			Read D	evice Variables	
🗉 🔛 Конфигурация задач		Cmn 34	34 Epsilon LD				× ariable Damping Val	lue
Regul Bus (Regul Bus)		_						
🖃 🕤 0 : R500_ (R500 Крейт)				_				
ST_xx_x1x (ST xx x1x)				Ожидаемый	і тип переменной: Hart	Device_Cmn_3	4	
PP_00_xxx (PP 00 xxx)								
*: CU_00_xxx (CU 00 xxx)								
AI_16_081 (AI 16 081)						OK		
Hart_Master (Hart Master)								
Hart_Outer_Slave (Har								
CP_04_011 (CP 04 011)								
∭ ST_xx_x2x (ST xx x2x)								

Рисунок 26 – Сообщение об ошибке при несовпадении типов переменных

ОБРАБОТКА ВЫПОЛНЕНИЯ КОМАНДЫ

Обработка выполнения команды по таймеру состоит из следующих шагов:

- проверьте поле статуса выполнения команды. Если статус равен CmdStatus.Ok, то команда была выполнена успешно и можно работать с полученными данными;
- ошибка в статусе команды означает сбой при выполнении команды либо наличие ошибки в самом ответе от устройства и требуется дополнительный анализ кода CmdStatus либо анализ полей команды CmdRespCode и CmdDevStatus.

Обработка выполнения команды по триггеру состоит из следующих шагов:

 для добавления команды в очередь выполнения по триггеру установите в коде программы для поля CmdTrigger команды значение True (предварительно это поле должно быть выставлено в False):

```
IF Условие_запуска_команды_на_выполнение = TRUE THEN

P_Sensor_cmd01.CmdTrigger := TRUE;

END IF
```

- отслеживайте успешное выполнение команды по полю статуса:
 - о если статус paвeн CmdStatus.Ok, то команда в последнем цикле была выполнена успешно и можно работать с полученными данными;
 - о ошибка в статусе команды означает сбой при выполнении команды либо наличие ошибки в самом ответе от устройства и требуется дополнительный анализ кода CmdStatus либо анализ полей команды CmdRespCode и CmdDevStatus.

Обработка полученных значений в поле данных команды:

Обработка выполнения Burst команды состоит из следующей последовательности действий:

- проверьте поле статуса выполнения команды. Если статус равен CmdStatus.Ok, то Burst пакет был обработан успешно и можно работать с полученными данными;
- ошибка в статусе команды CmdStatus.ErrorInResponse означает сбой при обработке Burst пакета, либо наличие ошибки в самом ответе от устройства, и требуется дополнительный анализ кода CmdStatus, либо анализ полей команды CmdRespCode и CmdDevStatus.



ИНФОРМАЦИЯ

При выполнении пользовательских команд используется «длинный» (5 байт) уникальный адрес устройства, который формируется в библиотеке при получении ответа на запрос команды идентификации (Universal Command #0). Поэтому, если уникальный адрес еще не сформирован, то перед выполнением пользовательской команды библиотека автоматически добавляет команду идентификации и пытается ее выполнить. Также эта команда будет автоматически выполняться в случае ошибки обмена с устройством перед следующей в очереди пользовательской командой

ПОДДЕРЖКА УСТРОЙСТВ В BURST РЕЖИМЕ

Burst режим позволяет устройству циклически отправлять в сеть оперативные данные без запроса со стороны мастера. Только одно устройство в сети может быть переведено в этот режим. Для настройки Burst режима используются команды, описанные в таблице 2:

Tagarran	L'an comment		110.000	a a virterr	Dunat	
таолина z –	команлы	лля	наст	роики.	DUISU	режима
1 400000	100000000000000000000000000000000000000	~~~		P • • • • • • •	2	p •

Команда	Описание
Command 109 Burst Mode Control	Включение и выключение Burst режима
Command 108 Write Burst Mode Command Number	Задает выбор номера команды, ответ на которую устройство будет транслировать в сеть. К таким командам относятся команды 1, 2, 3, 9 и 33. Команда может выполняться как на этапе конфигурации устройства, так и в процессе работы в режиме Burst «на лету»
Command 107 Write Burst Device Variables	Позволяет выбрать номера Device Variables в устройстве, которые будут переданы в ответе на команду 9 и 33
Command 105 Read Burst Mode Configuration	Чтение текущей конфигурации Burst

Для организации получения данных от устройства в Burst режиме, подключенном на один из каналов, необходимо выполнить следующие действия:

- 1. Добавьте HART-устройство на канал (HART Master ⇒ Hart Outer Slave). Установите в общих параметрах устройства флажок в поле Burst режим.
- 2. Добавьте контейнер HartDevice (Application ⇒Добавить объект... ⇒ HartDevice...). На вкладке Редактор hart устройства выберите hart устройство и импортируйте в контейнер необходимые для настройки (105,107,108,109) и получения данных (1,2,3,9,33) команды.
- 3. Перейдите на вкладку **Hart команды** объекта **Hart Outer Slave** (п.1). Нажмите кнопку **Э**. Активируйте созданный в п.2 контейнер команд, установив флажок в поле **Выбрать**. Далее установите флажок в поле **Burst режим** для тех команд, по которым будут приходить оперативные данные от устройства в Burst режиме.



ВНИМАНИЕ!

В контейнер команд **HartDevice** можно добавить только один экземпляр команды каждого конкретного типа. При этом команда может обрабатываться либо в обычном режиме (запрос/ответ, циклически или по триггеру), либо в Burst режиме (получение пакетов от Burst устройства в режиме прослушивания сети, только команды 1, 2, 3, 9, 33), в зависимости от состояния флажка команды из п.3. Тип вызова команд не важен, если выставлен флажок в поле Burst режим

4. Добавьте в код переменные соответствующих типов (PLC_PRG). В экземпляры этих переменных будут копироваться данные из полученных Burst пакетов. В списке команд из п.3 на вкладке **Hart команды** привяжите добавленные переменные к активным командам с помощью ассистента ввода в поле **Переменная для привязки**. (Рисунок 27).

Настройка На	rt OuterSlave	Hart команды 🚘	Regul Hart Outer S	lave IEC Objects Coc	тояние 🌗 Информация			
Hart устройство								
Привязанно	ре устройство:	HartDevice						
Выберите н	ювое hart устро	йство:						
HartDevice	1		× G	Ð				
Команды:								
Выбрать	Имя	Код команды	Тип вызова	Время цикла (мс)	Переменная для привязки	Описание	Burst режим	
	Uni_1	1	Timer	1000	PLC_PRG.dev1_cmd01	Read Primary Variable		
···· 🗹	Uni_2	2	Timer	1000	PLC_PRG.dev1_cmd02	Read Loop Current And Percent Of Range	\checkmark	
	Uni_3	3	Timer	1000	PLC_PRG.dev1_cmd03	Read Dynamic Variables And Loop Current	\checkmark	
···· 🗹	Uni_9	9	Timer	1000	PLC_PRG.dev1_cmd09	Read Device Variables with Status	\checkmark	
	Cmn_33	33	Timer	1000	PLC_PRG.dev1_cmd33	Read Device Variables	\checkmark	
···· 🗹	Cmn_105	105	Trigger	-	PLC_PRG.dev1_cmd105	Read Burst Mode Configuration		
	Cmn_107	107	Trigger	-	PLC_PRG.dev1_cmd107	Write Burst Device Variables		
···· 🗸	Cmn_108	108	Trigger	-	PLC_PRG.dev1_cmd108	Write Burst Mode Command Number		
i	Cmn_109	109	Trigger	-	PLC_PRG.dev1_cmd109	Burst Mode Control		

Рисунок 27 - Добавление и привязка команд для Burst режима

5. После загрузки и запуска приложения на контроллере экземпляр объекта Hart Outer Slave переходит в рабочий режим – выполнение обычных команд (запрос/ответ) и прослушивание сети на предмет обнаружения Burst пакетов. Уникальность команд в контейнере HartDevice, привязанном к объекту Hart Outer Slave, позволяет однозначно сопоставить экземпляр переменной МЭК-приложения и данные в полученном Burst пакете. В случае успешной обработки пакета, данные из него, включая байты Response Code и Field Device Status, копируются в соответствующие поля структуры переменной МЭК-приложения, а поле CmdStatus принимает значение Ok (3). В случае ошибки поле CmdStatus устанавливается в значение ErrorInResponse (5).

ФУНКЦИОНАЛЬНЫЙ БЛОК HARTUSERREQUEST И ДИНАМИЧЕСКИ ФОРМИРУЕМАЯ КОМАНДА

Общее описание

В библиотеке имеется возможность работы в МЭК-приложении с динамически формируемой командой HART, реализованной в функциональном блоке (далее – ФБ) *HartUserRequest*. Работа с командой – только по триггеру. Burst команды не поддерживаются.

Менеджер библиотек								x			
🏝 Добавить библиотеку 🗙 Уда	алить библиотеку 🛛 🚰 С	войства 🗃 Детали 📑	3an	олни	птел	и 👔 Репозиторий библ	пиотек				
Имя						Дополнительное имя	Действующая версия	^			
B	(Prosoft-Systems Ltd)					PsIoDrvHartMaster	1.5.0.4				
						DeCeE	1 - 1 1	×			
🖃 🎒 iodrvpshartmaster 🔍 💌	Входы/Выходы Графич.	Документация									
helpers	FUNCTION_BLOCK HartUserRequest										
💼 📑 DefaultHartDeviceRev6	Имя	Тип	н	Α.	н	Комментарий					
HartUserRequest	₩ xExecute	BOOL				М: Управление работой Ф	ÞБ				
	★ refHartOuterSlave	REFERENCE TO HartSlave				М: Ссылка на опрашивае	мое устройство				
	🍬 byCommand	BYTE				М: Код команды					
	byRepeatNum	BYTE				М: Число повторных запр	росов				
	🍬 sSpecificName	STRING			Ή	О: Имя запроса					
	🐐 pbyRDataFormat	POINTER TO BYTE				М: Указатель на формат,	т данных для приема				
	byRDataFormatLen	BYTE				М: Размер массива форма) массива формата данных для приема в байтах				
	byXDataFormat POINTER TO BYTE					М: Указатель на формат,	данных для передачи				
	byXDataFormatLen	BYTE			М: Размер массива формата данных для передачи в байтах			ах			
	🍬 pbyRData	POINTER TO BYTE				М: Буфер данных для приема					
	🐐 byRDataLen	BYTE				М: Размер буфера данны:					
	👋 pbyXData	POINTER TO BYTE				М: Буфер данных для пер					
	🔖 byXDataLen	BYTE				М: Размер буфера данны:	х для передачи в байтах				
	🍫 xDone	BOOL				Запрос выполнен успешно					
	🍫 xError	BOOL				Ошибка выполнения запр					
	🍫 eStatus	CmdStatus				Код ошибки					
	🍫 bRespCode	BYTE				Код ответа					
	🍫 bDevStatus	BYTE				Статус устройства					
< >											

Рисунок 28 – Функциональный блок HartUserRequest

Для корректной обработки произвольной HART-команды в переменную ФБ *HartUserRequest* при ее вызове в МЭК-приложении передаются специфические параметры, включающие форматы структур поля данных для записи (в запросе) и чтения (в ответе). Это позволяет библиотеке корректно отформатировать выходной/входной блок данных запроса/ответа и сопоставить разнотипные параметры, содержащиеся в этом блоке, соответствующим полям переменной команды, объявленной в МЭК-приложении.

Ниже приведен пример выполнения универсальной команды запроса информации об устройстве (команда с кодом 15). При этом доступны два варианта задания структуры данных.

Объявление структуры данных команды непосредственно в коде

Для объявления структуры данных команды непосредственно в коде необходимо наследовать структуру 15-й команды от базовой структуры CmdBase, объявленной в библиотеке (Рисунок 29).

```
//UNI 15 - Read Device Information;
TYPE cmd15 EXTENDS PsIoDrvHartMaster.CmdBase :
STRUCT
           data : cmd15 rdata; //UNI 15 - Read Device Information;
END STRUCT
END TYPE
```

Ленеджер библиотек					
🛛 Добавить библиотеку 🗙 Удали	ть библиотеку 🛛 📩	Свойства 🧃	🕽 Детали	写 Заполнители	
1мя					Дополнительное имя
🕬 PsIoDrvHartMaster, 1.5.0.4 (Pro	osoft-Systems Ltd)				PsIoDrvHartMaster
					>
iodrvpshartmaster	Входы/Выходы	Цокументация			
÷ 🚗 –	STRUCT CHILDR	ise			
Types	Имя	Тип	F a 1	Комментарий	
Types	Имя	Тип ВҮТЕ	F J I	Комментарий Communication Stat	us/ResponseCode;
Types CmdBase CmdBase CmdStatus DefaultHartDeviceRev6	Имя	Тип BYTE BYTE	H J I	Комментарий Communication Stat Field Device Status	us/ResponseCode; ;
Types CmdBase CmdBase CmdStatus DefaultHartDeviceRev6 HartUserRequest	Имя	Тип ВYTE ВYTE CmdStatus	F J 1	Комментарий Communication Stat Field Device Status Command execute s	us/ResponseCode; ; ;tatus;

Рисунок 29 – Базовая структура CmdBase

С полем данных типа структуры cmd15 rdata, соответствующей спецификации HART.

```
{attribute 'pack mode' := '1'}
TYPE cmd15 rdata :
                                                                                                      //UNI 15 - Read Device Information;
STRUCT
                FPV_AlarmSelCode : BYTE;PV_TransFuncCode : BYTE;PV_UTransFuncCode : BYTE;PV_UL_UnitCode : BYTE;PV_Upper and Lower Range Values Units Code;PV_UpperRangeValue: REAL;PV_LowerRangeValue: REAL;PV_LowerRangeValue: REAL;PV_DampingValue : REAL;PV_DampingValue : REAL;PV_DampingValue : REAL;PV_DampingValue : REAL;PV_Damping Value;WriteProtectCode : BYTE;PrivLblDistrCode : BYTE;PV_AnalogChFlags : BYTE;PV_PV_Analog Channel Flags;
END STRUCT
END TYPE
```

Далее, в области объявлений программного блока (POU) обработки HART-команды следует объявить набор рабочих переменных.

```
//экземпляр FB пользовательского запроса;
UserRequest : PsIoDrvHartMaster.HartUserRequest;
//код команды;
bUR_Command : BYTE := 0;
```

```
//число попыток выполнения команды;
     bUR RepeatNum : BYTE := 4;
     //выходной параметр ФБ - флаг успешного выполнения команды;
     xUR Done : BOOL := FALSE;
     //выходной параметр ФБ - флаг ошибки выполнения команды;
     xUR Error : BOOL := FALSE;
     //размер структуры данных ответа;
     bUR CommandRxDataLen: BYTE;
     //размер структуры данных запроса;
     bUR CommandTxDataLen: BYTE;
     //флаг управления запуска/остановки команды;
     xUR Execute Uni 15 : BOOL := FALSE;
     //структура команды;
     cmdUR Command 15 : cmd15;
      //формат блока данных ответа от устройства, содержащий размер полей данных в
байтах;
     arrbUR CommandRxFrm 15: ARRAY[0..9] OF BYTE := [1, 1, 1, 4, 4, 4, 1, 1, 1,
0];
     //количество полей блока данных ответа, включая завершающий '0';
     bUR CommandRxFrmtLen 15: BYTE := 10;
     //формат блока данных запроса к устройству, содержащий размер полей данных в
байтах;
     arrbUR CommandTxFrmt 15: POINTER TO BYTE; //для этой команды не используется;
     //количество полей блока данных запроса, включая завершающий '0';
     bUR CommandRTxFrmtLen 15: BYTE;
                                                           //для этой команды не
используется;
```

В теле блока обработка команды сводится к следующим строкам:

```
// Выполнение пользовательского запроса
IF xUR Execute_Uni_15 = TRUE THEN
     bUR Command := 15;
     bUR CommandRxDataLen := UINT TO BYTE(SIZEOF(cmdUR Command 15.data));
     bUR CommandTxDataLen := 0;
     UserRequest(
                    xExecute := xUR Execute Uni 15,
                 refHartOuterSlave := Hart Outer Slave,
                 byCommand := bUR Command,
                 byRepeatNum := bUR RepeatNum,
                 sSpecificName := 'Test',
                 pbyRData := ADR(cmdUR Command 15.data),
                 byRDataLen := bUR CommandRxDataLen,
                 pbyRDataFormat := ADR(arrbUR CommandRxFrm 15),
                 byRDataFormatLen := bUR CommandRxFrmtLen 15,
                 bRespCode => cmdUR Command 15.RespCode,
                 bDevStatus => cmdUR Command 15.DevStatus,
                 eStatus => cmdUR Command 15.Status,
                 xDone => xUR Done,
                 xError => xUR Error
     );
     IF xUR Done OR xUR Error THEN
           xUR Execute Uni 15 := FALSE;
           UserRequest(
                           xExecute := xUR Execute Uni 15);
     END IF
END IF
```

В данном случае Hart Outer Slave – это имя экземпляра функционального блока, соответствующего устройству Hart Outer Slave в дереве устройств.

Использование команды из контейнера команд HART-устройства

Для использования команды из контейнера команд HART-устройства необходимо создать в дереве устройств фиктивный контейнер и добавить/импортировать в него 15-ю команду, для которой автоматически создается тип структуры HartUserRequest Uni 15 (Рисунок 30).

Устройства 👻 🕂 🗙	📔 Редактор hart команды 🗙 📃					
Hart_project Art_project Gild REGUL_R500_71 (REGUL R500-71 Gild REGUL_R500_71 (REGUL R500-71	Параметры команды Имя: Uni_15	Проверка формата структур Результат: Ок				
Application	Код: 15		Требуемый формат структуры передачи: [0] Требуемый формат структуры приёма: [1,1,1,4,4,4,1,1,1,0]			
Гановенсериент □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	Описание: Read Device Information Формат структуры передачи данных	Формат структуры	приёма данных			
РС_PRG В - 🥵 Конфигурация задач	Добавить новое поле	Добавить новое пол	ь новое поле			
E Regui Bus (Regui Bus)	Имя Тип Размер Описание	Имя		Тип Ра	азмер Описание	
		PV_Alarm_Selection	on_Code	BYTE 1	PV Alarm Selection Code	
		PV_Transfer_Funct	tion_Code	BYTE 1	PV Transfer Function Code	
		PV_Upper_and_Lo	ower_Range_Values_Units_Code	BYTE 1	PV Upper and Lower Range	
		PV_Upper_Range	_Value	REAL 4	PV Upper Range Value	
		PV_Lower_Range_	Value	REAL 4	PV Lower Range Value	
		PV_Damping_Valu	ie	REAL 4	PV Damping Value	
		Write_Protect_Co	de	BYTE 1	Write Protect Code	
< >		Private_Label_Dis	tributor_Code	BYTE 1	Private Label Distributor Co	
🛫 Устройства 🗋 POU		PV_Analog_Chann	el_Flags	BYTE 1	PV Analog Channel Flags	
		Последня	ія компиляция: 😳 0 😷 2 — П	редкомпил.:	Текущий пользова	

Рисунок 30 – Структура HartUserRequest Uni 15

Далее, в области объявлений программного блока (POU) обработки HART-команды следует объявить набор рабочих переменных.

```
//экземпляр FB пользовательского запроса;
     UserRequest
                   : PsIoDrvHartMaster.HartUserRequest;
     //код команды;
     bUR Command : BYTE := 0;
     //число попыток выполнения команды;
     bUR RepeatNum : BYTE := 4;
     //выходной параметр ФБ - флаг успешного выполнения команды;
     xUR Done
                           : BOOL := FALSE;
     //выходной параметр ФБ - флаг ошибки выполнения команды;
     xUR Error : BOOL := FALSE;
     //размер структуры данных ответа;
     bUR CommandRxDataLen: BYTE;
     //размер структуры данных запроса;
     bUR CommandTxDataLen: BYTE;
     //структура команды;
     cmd 15 : HartUserRequest Uni 15;
     //формат блока данных ответа от устройства, содержащий размер полей данных в
байтах;
     arrbUR CommandRxFrm 15: ARRAY[0..9] OF BYTE := [1, 1, 1, 4, 4, 4, 1, 1, 1,
01;
     //количество полей блока данных ответа, включая завершающий '0';
     bUR CommandRxFrmtLen 15: BYTE := 10;
     //формат блока данных запроса к устройству, содержащий размер полей данных в
байтах;
     arrbUR CommandTxFrmt 15: POINTER TO BYTE; //для этой команды не используется;
     //количество полей блока данных запроса, включая завершающий '0';
     bUR CommandRTxFrmtLen 15: BYTE;
                                            //для этой команды не используется;
```

В теле блока обработка команды сводится к следующим строкам:

```
// Выполнение пользовательского запроса
IF cmd 15.CmdTrigger = TRUE THEN
     bUR Command := 15;
     bUR CommandRxDataLen := UINT TO BYTE(SIZEOF(cmd 15.rx data));
      bUR CommandTxDataLen := 0;
                      xExecute := cmd 15.CmdTrigger,
      UserRequest (
                 refHartOuterSlave := Hart Outer Slave,
                 byCommand := bUR Command,
                 byRepeatNum := bUR RepeatNum,
                  sSpecificName := 'Test',
                  pbyRData := ADR(cmd_15.rx_data),
                 byRDataLen := bUR_CommandRxDataLen,
                  pbyRDataFormat := ADR(arrbUR CommandRxFrm 15),
                 byRDataFormatLen := bUR CommandRxFrmtLen 15,
                 bRespCode => cmd 15.CmdRespCode,
                 bDevStatus => cmd 15.CmdDevStatus,
                  eStatus => cmd 15.CmdStatus,
                  xDone => xUR Done,
                 xError => xUR Error
      );
      IF xUR Done OR xUR Error THEN
            cmd 15.CmdTrigger := FALSE;
            UserRequest( xExecute := cmd 15.CmdTrigger);
     END IF
END IF
```

Пример выполнения команды 34 – Write Primary Variable Damping Value

Для объявления структуры данных команды непосредственно в коде необходимо наследовать структуру 34-й команды от базовой структуры CmdBase, объявленной в библиотеке.

```
//COMMON PRACTICE 34 - Write Primary Variable Damping Value;
TYPE cmd34 EXTENDS PsIoDrvHartMaster.CmdBase:
STRUCT
     xdata : cmd34 data;
                           //COMMON PRACTICE 34 - OUT - Write Primary Variable
Damping Value;
    rdata : cmd34 data; //COMMON PRACTICE 34 - IN - Write Primary Variable
Damping Value;
END STRUCT
END TYPE
```

Где:

```
{attribute 'pack mode' := '1'}
TYPE cmd34 data : //COMMON PRACTICE 34 - Write Primary Variable Damping Value;
STRUCT
      PV DampungValue : REAL;//Primary Variable Damping Value (units of seconds);
END STRUCT
END TYPE
```

Далее, в области объявлений программного блока (POU) обработки HART-команды следует объявить набор рабочих переменных.

```
//экземпляр FB пользовательского запроса;
     UserRequest
                                 : PsIoDrvHartMaster.HartUserRequest;
     //код команды;
```

```
bUR Command
                                  : BYTE := 0;
     //число попыток выполнения команды;
     bUR RepeatNum : BYTE := 4;
     //выходной параметр ФБ - флаг успешного выполнения команды;
                               : BOOL := FALSE;
     xUR Done
     //выходной параметр ФБ - флаг ошибки выполнения команды;
                               : BOOL := FALSE;
     xUR Error
     //размер структуры данных ответа;
     bUR CommandRxDataLen: BYTE;
     //размер структуры данных запроса;
     bUR CommandTxDataLen: BYTE;
     //флаг управления запуска/остановки команды;
     xUR Execute Cmn 34 : BOOL := FALSE;
     //структура команды;
     cmdUR Command 34 : cmd34;
     //формат блока данных ответа от устройства, содержащий размер полей данных в
байтах;
     arrbUR CommandRxFrmt 34: ARRAY[0..1] OF BYTE := [4, 0];
     //количество полей блока данных ответа, включая завершающий '0';
     bUR CommandRxFrmtLen 34: BYTE := 2;
     //формат блока данных запроса к устройству, содержащий размер полей данных в
байтах;
     arrbUR CommandTxFrmt 34: ARRAY[0..1] OF BYTE := [4, 0];
     //количество полей блока данных запроса, включая завершающий '0';
     bUR CommandTxFrmtLen 34: BYTE := 2;
```

В теле блока обработка команды сводится к следующим строкам:

```
IF xUR_Execute_Cmn_34 = TRUE THEN
     bUR Command := 34;
     bUR_CommandRxDataLen := UINT_TO_BYTE(SIZEOF(cmdUR_Command_34.rdata));
     bUR CommandTxDataLen := UINT TO BYTE(SIZEOF(cmdUR Command 34.xdata));
     cmdUR Command 34.xdata.PV DampungValue := 4.0;
                      xExecute := xUR Execute Cmn 34,
     UserRequest (
                 refHartOuterSlave := Hart Outer Slave,
                 byCommand := bUR Command,
                 byRepeatNum := bUR RepeatNum,
                 sSpecificName := 'Test',
                 pbyRData := ADR(cmdUR Command 34.rdata),
                 byRDataLen := bUR CommandRxDataLen,
                 pbyRDataFormat := ADR(arrbUR CommandRxFrmt 34),
                 byRDataFormatLen := bUR CommandRxFrmtLen 34,
                 byXDataLen := bUR CommandTxDataLen,
                 pbyXData := ADR (cmdUR Command 34.xdata),
                 pbyXDataFormat := ADR (arrbUR CommandTxFrmt 34),
                 byXDataFormatLen := bUR CommandTxFrmtLen 34,
                 bRespCode => cmdUR Command 34.RespCode,
                 bDevStatus => cmdUR Command 34.DevStatus,
                 eStatus => cmdUR Command 34.Status,
                 xDone => xUR Done,
                 xError => xUR Error
     );
     IF xUR Done OR xUR Error THEN
           xUR Execute Cmn 34 := FALSE;
           UserRequest( xExecute := xUR_Execute_Cmn_34);
     END IF
END IF
```